Analysis and Segmentation of Face Images using Point Annotations and Linear Subspace Techniques

Mikkel B. Stegmann

Informatics and Mathematical Modelling, Technical University of Denmark
Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, Denmark
IMM Technical Report IMM-REP-2002-22, August 2002*

Abstract

This report provides an analysis of 37 annotated frontal face images. All results presented have been obtained using our freely available Active Appearance Model (AAM) implementation. To ensure the reproducibility of the presented experiments, the data set has also been made available. As such, the data and this report may serve as a point of reference to compare other AAM implementations against. In addition, we address the problem of AAM model truncation using parallel analysis along with a comparable study of the two prevalent AAM learning methods; principal component regression and estimation of fixed Jacobian matrices. To assess applicability and efficiency, timings for model building, warping and optimisation are given together with a description of how to exploit the warping capabilities of contemporary consumer-level graphics hardware.

Keywords: shape analysis, generative modelling, face recognition, active shape models, active appearance models, annotated image data set.

Contents

*Updated August 2003.

1 Introduction

Face images are of particular interest to many within image analysis. Aside from the many interesting applications – such as biometric authentication, video-assisted speech recognition, low-bandwidth video conferencing, eye-tracking, database indexing, digital puppeteering et cetera – face images posses one quality that separate them from all other object classes. We are all experts in interpreting face images. Even very subtle changes to a face are easy for us to detect. And has been for a long time. Shortly after birth, infants are able to recognise the faces of their mother and father and associate these with fulfilment and security. Soon after, skills for evaluating facial expressions are developed. This is why we like face images so much when evaluating and presenting image analysis algorithms.

Figure 1: Expert in face interpretation.

This report presents an analysis of a set of annotated face images. The foundation for this analysis is the Active Appearance Models (AAMs) [8, 1] of Manchester University.

We have aimed at exposing as much as possible of the involved data structures in the generation of facial AAMs, which renders this report somewhat pictorial. Further, in the spirit of reproducible research the data set is made available for download. As such this report may serve as a point of reference to compare other AAM implementations against. The intended audience is researchers and students familiar with the AAM framework.[1] For those that are not, an introduction to AAMs is given. In that sense, we hope that the report also will pass on valuable information to readers solely interested in analysis of faces.

Due to the absence of a comparison between the two prevalent AAM training methods, we give results using both.[2] Further, we treat the problem of selecting an "appropriate" number of model modes. We call this *model truncation* in the following. Most of the analyses are carried out in both grey-scale and colour and at two different scales.

The report is organised as follows. Section 2 describes the data material. Section 3 gives a brief introduction to AAMs and treats the aspects of training and model truncation. Section 4 describes issues regarding the implementation. Section 5 presents the experimental results, while Section 6 serves a discussion and draws some concluding remarks.

2 Data Material

The data set comprises 37 still images of 37 different frontal human faces, all without glasses and with a neutral expression. The gender distribution is 7 females and 30 males. Images were acquired in 640×480 JPEG colour format using a Sony DV video camera (DCR-TRV900E PAL) and subsequently converted to the BMP image file format. The following facial structures were manually annotated using 58 landmarks: eyebrows, eyes, nose, mouth and jaw. A total of seven point paths were used; three closed and four

[1] Or familiar with similar (sub)designs such as Eigen-faces [15], Active Shape Models [6], Active Blobs [13], Morphable Models [11] et cetera.

[2] Simultaneously with the first revision of this report, a comparable study was published in [3], which is not refered in this report. Pleae consult [3] to compare the results given here.

open. All annotations were formatted in ASF, which is described in appendix A. Refer to Figure 2 for an example annotation. All face images are shown in Appendix C.

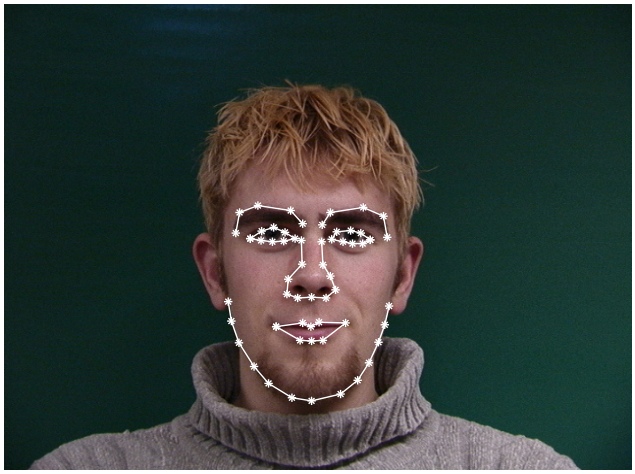


Figure 2: Example annotation of a face using 58 landmarks.

2.1 Terms of Use

The dataset can be used freely for education and research when this report is cited. For convenience the corresponding BibTeX entry is given below.

```
@TECHREPORT{StegmannFaceRep2002,
    author       = {Stegmann, M. B.},
    title        = {Analysis and Segmentation of Face Images using
                    Point Annotations and Linear Subspace Techniques},
    year         = {2002},
    institution  = {Informatics and Mathematical Modelling, Technical University of Denmark, DTU},
    address      = {Richard Petersens Plads, Building 321, DK-2800 Kgs.\ Lyngby},
    url          = {http://www.imm.dtu.dk/~aam/},
    isbn_issn    = {IMM-REP-2002-xx}
}
```

2.2 Obtaining the Data Material

A package containing images and their corresponding annotations can be obtained from:

`http://www.imm.dtu.dk/~aam/`

The package size is approximately 22 MB and in zip format.

3 Active Appearance Models

Active Appearance Models establish a compact parameterisation of object variability, as learned from a training set by estimating a set of latent variables. The modelled object properties are usually shape and pixel intensities. The latter is henceforward denoted *texture*. From these quantities new images similar to the training set can be generated.

Objects are defined by marking up each example with points of correspondence (i.e. landmarks) over the set either by hand, or by semi- to completely automated methods. The key to the compactness of these models lies in proper compensation of shape variability prior to modelling texture variability. Models failing in doing this, such as the Eigen-face model [15], experience difficulties in modelling variability in a compact manner.

Exploiting prior knowledge about the local nature of the optimisation space, these models can be rapidly fitted to unseen images, given a reasonable initialisation.

Variability is modelled by means of a Principal Component Analysis (PCA), i.e. an eigen analysis of the dispersions of shape and texture. Let there be given P training examples for an object class, and let each example be represented by a set of N landmark

points and M texture samples. The shape examples are aligned to a common mean using a Generalised Procrustes Analysis (GPA) [9]. The Procrustes shape coordinates are subsequently projected into the tangent plane to the shape manifold, at the pole given by the mean shape. The texture examples are warped into correspondence using a piecewise affine warp and subsequently sampled from this *shape-free* reference. Typically, this geometrical reference frame is the Procrustes mean shape. Let $\mathbf{s}$ and $\mathbf{t}$ denote a synthesized shape and texture and let $\bar{\mathbf{s}}$ and $\bar{\mathbf{t}}$ denote the corresponding sample means. New instances are now generated by adjusting the PC scores, $\mathbf{b}_s$ and $\mathbf{b}_t$ in

$$\mathbf{s} = \bar{\mathbf{s}} + \mathbf{\Phi}_s \mathbf{b}_s \quad , \quad \mathbf{t} = \bar{\mathbf{t}} + \mathbf{\Phi}_t \mathbf{b}_t \tag{1}$$

where $\mathbf{\Phi}_s$ and $\mathbf{\Phi}_t$ are eigenvectors of the shape and texture dispersions estimated from the training set. To obtain a combined shape and texture parameterisation, $\mathbf{c}$, the values of $\mathbf{b}_s$ and $\mathbf{b}_t$ over the training set are combined into

$$\mathbf{b} = \begin{bmatrix} \mathbf{W}_s \mathbf{b}_s \\ \mathbf{b}_t \end{bmatrix} = \begin{bmatrix} \mathbf{W}_s \mathbf{\Phi}_s^{\mathrm{T}} (\mathbf{s} - \bar{\mathbf{s}}) \\ \mathbf{\Phi}_t^{\mathrm{T}} (\mathbf{t} - \bar{\mathbf{t}}) \end{bmatrix}. \tag{2}$$

A suitable weighting between pixel distances and pixel intensities is carried out through the diagonal matrix $\mathbf{W}_s$. To recover any correlation between shape and texture the two eigenspaces are usually coupled through a third PC transform

$$\mathbf{b} = \mathbf{\Phi}_c \mathbf{c} \tag{3}$$

obtaining the combined appearance model parameters, $\mathbf{c}$, that generate new object instances by

$$\mathbf{s} = \bar{\mathbf{s}} + \mathbf{\Phi}_s \mathbf{W}_s^{-1} \mathbf{\Phi}_{c,s} \mathbf{c} \quad , \quad \mathbf{t} = \bar{\mathbf{t}} + \mathbf{\Phi}_t \mathbf{\Phi}_{c,t} \mathbf{c} \quad , \quad \mathbf{\Phi}_c = \begin{bmatrix} \mathbf{\Phi}_{c,s} \\ \mathbf{\Phi}_{c,t} \end{bmatrix}. \tag{4}$$

To regularise the model and improve speed and compactness, $\mathbf{\Phi}_s$, $\mathbf{\Phi}_t$ and $\mathbf{\Phi}_c$ are truncated, usually such that a certain amount of variance in the training set is preserved. This eventually results in k ($k < P$) combined modes, i.e. k dynamic parameters encoded in the vector $\mathbf{c}$.

The object instance, $(\mathbf{s}, \mathbf{t})$, is synthesised into an image by warping the pixel intensities of $\mathbf{t}$ into the geometry of the shape $\mathbf{s}$. Given a suitable similarity measure the model is matched to an unseen image using an iterative updating scheme based on a fixed Jacobian estimate [2, 4] or a principal component regression [1].

This sums up the basic theory of AAMs. For further details refer to [1, 2, 4, 14].

3.1 Model Training

Traditionally, AAMs have been trained to update model and pose parameters using one of two schemes described in the follwing. These parameters updates are carried using difference images between the current model image and the corresponding part of the unseen image that it covers. Applying such parameter corrections in an iterative scheme should thus drive the model towards the ground truth shape in the image.

Multivariate Regression

The initial AAM formulation use a regression approach where difference vectors, $\delta\mathbf{t} = \mathbf{t}_{image} - \mathbf{t}_{model}$, are regressed onto corresponding parameter perturbation/displacement vectors, $\delta\mathbf{p}$. Here $\mathbf{p}$ is either model or pose parameters, having the length Q. The goal is thus to obtain an optimal – in a least-squares sense – prediction matrix, $\mathbf{R}$, satisfying the linear relationship:

$$\delta\mathbf{p} = \mathbf{R}\delta\mathbf{t}. \tag{5}$$

Let there be conducted S perturbation experiments and let

$$\mathbf{P} = \begin{bmatrix} \vdots & & \vdots \\ \delta\mathbf{p}_1 & \dots & \delta\mathbf{p}_S \\ \vdots & & \vdots \end{bmatrix} \quad \text{and} \quad \mathbf{T} = \begin{bmatrix} \vdots & & \vdots \\ \delta\mathbf{t}_1 & \dots & \delta\mathbf{t}_S \\ \vdots & & \vdots \end{bmatrix}. \tag{6}$$

During these perturbation experiments $\delta\mathbf{t}_i$ is short for $\delta\mathbf{t}_i(\mathbf{p}^+ + \delta\mathbf{p}_i)$, where $\mathbf{p}^+$ denotes the optimal parameter configuration. Since $Q \ll S \ll M$ typically, $\mathbf{R}$ is estimated using principal component regression [4]. First thing to notice is the rather large matrix $\mathbf{T}$. From this an s-dimensional subspace ($s \leq S$) is extracted. This makes $\mathbf{R}$ well-determined in $\mathbf{P} = \mathbf{R}\mathbf{T}'$, where $\mathbf{T}'$ contains s-dimensional projected versions of $\mathbf{T}$. Consequently, an eigen-decomposition of an $S \times S$ matrix is involved. To keep this matrix down to a feasible size, typically only a subset of the training set is used for training. This is especially important when estimating $\mathbf{R}_c$, as the number of model modes grows with the number of examples, P. For models with a small number of training examples the growth in S becomes close to quadratic. In the experiments below the same training subset has been used to estimate both $\mathbf{R}_c$ and $\mathbf{R}_t$.

First Order Taylor Approximation

In later AAMs publications [2, 5, 7] the multivariate regression is superseded by a simpler approach. It is easier to implement, faster to calculate and requires far less memory to execute. Further, [2] claims that it is "more reliable". The approach is derived as follows. First we introduce the residual vector $\mathbf{r}$, parameterised by $\mathbf{p}$:

$$\mathbf{r}(\mathbf{p}) = \delta\mathbf{t}(\mathbf{p}) = \mathbf{t}_{image}(\mathbf{p}) - \mathbf{t}_{model}(\mathbf{p}). \tag{7}$$

A first order Taylor expansion of $\mathbf{r}$ at $\mathbf{p}^*$ is

$$\mathbf{r}(\mathbf{p}^* + \delta\mathbf{p}) \approx \mathbf{r}(\mathbf{p}^*) + \frac{\partial\mathbf{r}(\mathbf{p}^*)}{\partial\mathbf{p}}\delta\mathbf{p} \tag{8}$$

where $\mathbf{p}^*$ is in the proximity of $\mathbf{p}^+$ and

$$\frac{\partial\mathbf{r}(\mathbf{p}^*)}{\partial\mathbf{p}} = \frac{\partial\mathbf{r}}{\partial\mathbf{p}} = \begin{bmatrix} \frac{\partial r_1}{\partial p_1} & & \frac{\partial r_1}{\partial p_Q} \\ \vdots & \cdots & \vdots \\ \frac{\partial r_M}{\partial p_1} & & \frac{\partial r_M}{\partial p_Q} \end{bmatrix}. \tag{9}$$

The goal of the parameter update is to drive the residual vector to zero, i.e. finding $\mathbf{p}^+$. Using the L_2-norm the optimal $\delta\mathbf{p}$ is: $\arg\min_{\delta\mathbf{p}} |\mathbf{r}(\mathbf{p}^* + \delta\mathbf{p})|^2$. Hence, the least-squares solution of (8) becomes:

$$\delta\mathbf{p} = -\left(\frac{\partial\mathbf{r}}{\partial\mathbf{p}}^{\mathrm{T}}\frac{\partial\mathbf{r}}{\partial\mathbf{p}}\right)^{-1}\frac{\partial\mathbf{r}}{\partial\mathbf{p}}^{\mathrm{T}}\mathbf{r}(\mathbf{p}^*) = -\mathbf{R}\mathbf{r}(\mathbf{p}^*). \tag{10}$$

To obtain good numerical stability one would use a singular value decomposition (SVD) of the Jacobian, $\frac{\partial\mathbf{r}}{\partial\mathbf{p}}$, to obtain its pseudo-inverse, $\mathbf{R}$. However due to the size this is not feasible, why a normal matrix inversion must be carried out.

Normally, the Jacobian must be recalculated at each optimisation step, which is a very expensive task due to its size. However, since AAMs operate in a standardised domain, i.e. the shape free reference frame, AAMs perform the following approximation

$$\frac{\partial\mathbf{r}(\mathbf{p}^*)}{\partial\mathbf{p}} \approx \frac{\partial\mathbf{r}(\mathbf{p}^+)}{\partial\mathbf{p}}. \tag{11}$$

Further – and this is a somewhat crude assumption – the right-hand side of (11) is considered constant over all training examples. Thus, $\mathbf{R}$ is considered fixed and estimated once during the model building process using numeric differentiation on the training set [2]. In the subsequent sections, we call this learning approach the *Jacobian*.

Perturbation Scheme

The remaining design choice in both learning methods is the perturbation scheme. Typically, this – very important step in crafting a good AAM – is often not described (fully) in the literature.

In all experiments below we have used the perturbation scheme shown in Table 1. Each parameter was displaced, while the remaining parameters were set to zero. Pose and model parameters were treated independently resulting in two prediction matrices: $\mathbf{R}_c$ and $\mathbf{R}_{pose}$.

Table 1: Perturbation scheme used in both learning methods.

Variable	Perturbations
x, y	$\pm 5\%$, $\pm 10\%$ of the width and height of the reference shape, respectively
θ	± 5, ± 15 degrees
$scale$	$\pm 5\%$, $\pm 15\%$
c_{1-k}	± 0.25, ± 0.50 standard deviations

3.2 Model Truncation

In Section 3 we have truncated the eigenspaces of shape, texture and combined variation so that each explain a fixed amount of variance. Since the variance along the i-th principal axis is equal to the corresponding eigenvalue, λ_i this is easily carried out. To retain p percent of the variation, t modes can be chosen satisfying:

$$\sum_{i=1}^{t} \lambda_i \geq \frac{p}{100} \sum \lambda_i \tag{12}$$

A common assumption is to consider the remaining 5% of the signal to be noise. However, this is a classic bias/variance problem. Choosing a high value for p could result in a model too flexible, in the sense that it will fit to noise present in the training set (high variance) and thus not generalise very well. On the other hand, low p values would produce very restrictive models, not fitting the training set very well (high bias towards the model).

One method for choosing p is cross-validation, i.e. partitioning of the data set into a training and an independent test set. By adding modes the reconstruction error will initially decrease. But at some p the model will start to overfit the training set and the reconstruction error on the test set will start to increase. To make this procedure more robust several partitions can be used going towards a leave-one-out analysis in the extreme. More advanced methods such as the bootstrap could also be applied.

However, using cross-validation is quite cumbersome. A convenient alternative is parallel analysis introduced by Horn [10], where the data is compared to either i) normal distributed synthetic samples with a diagonal covariance matrix, or ii) a randomised version of the data matrix.[3] We will concentrate on the latter since it is imposing less strict distribution assumptions on the data. Further, it is simpler to implement and calculate.

In short, parallel analysis seeks to find the amount of eigenvalue inflation due to sampling errors (also known as *chance capitalisation* or *chance correlation*). In the perturbation version of parallel analysis this is estimated by breaking the correlation between variables, i.e. each variable is scrambled between observations. Using our column vector notation, this will be a randomisation of each row in the data matrix. The eigenvalues of this scrambled data matrix indicates the anisotropy of a data cloud, which is considered noise compared to the original data. From a scree plot of the eigenvalues of the original data and the scrambled data, modes that have higher eigenvalues for the scrambled than the unscrambled data, can thus be considered noise. The rationale is that these modes are only stemming from chance capitalisation and not actual population correlation.

Thus, for the very dense and highly correlated data in the AAM texture models, we will expect the eigenvalues stemming from parallel analysis to be very different from the data eigenvalues. Consequently, the opposite is expected for the sparser – and somewhat less correlated – shape models. This is not a characteristic behaviour of texture and

[3]The original work of John L. Horn used the former method.

shape models in themselves, but merely stemming from AAMs ability to recover dense correspondences with a sparse set of landmarks. Paulsen et al. [12] e.g., use very dense shape models and therefore experience scree plots with very different shape-eigenvalues.

Since the combined PCA in AAMs is done on the principal scores of the shape and texture models, variables have little correlation. This will result in nearly identical scree plots for the data and the randomised data. Thus only variance-based truncation is used on the combined models.

This data permutation version of Horn's parallel analysis is typically embedded in a Monte Carlo simulation scheme, where mean eigenvalues of several scrambling experiments is used to reduce sampling errors. These experiments are done with replacement due to the massive effort involved in keeping track of permutations.

4 Implementation

All subsequent experiments were carried out using an open source Active Appearance Model implementation. This is the AAM-API; a C++ implementation running under Microsoft Windows. A beta version of the AAM-API can be downloaded from the homepage mentioned in section 2.2.

AAMs rely heavily on image warps. In the current implementation, this can be carried out in software or by exploiting available OpenGL compliant graphics hardware. The latter is described in detail in Appendix B.

Where the results below are not obtained directly from the AAM-API, very basic Matlab scripts were used. All of these had no parameters. Hence, this should not limit the reader from reproducing the results.

5 Experimental results

This section aims at giving an in-depth – and somewhat pictorial – analysis of the described face images and their annotations. First the shape, texture and combined models are presented. Then the parameter prediction ability is tested followed by a set segmentation experiments. Finally the presented method for model truncation is tested and some machinery and timing details are given. Face observations are referred to through numbers from 1–37 that are obtained by an alphanumeric sort of the corresponding image filenames. If nothing is explicitly mentioned the results stem from the colour images.

5.1 Shape Model

The foundation of the shape model is the 58 facial landmarks shown in Figure 3 (left). To establish a reference coordinate system relative to these landmarks – and within their convex hull – a Delaunay triangulation is calculated in Figure 3 (right). This coordinate system is used later in the texture model. Notice the unpleasant triangles at the top.

Plotting the scatter of all 37 face annotations yields the rather confusing plot in Figure 4 (left). To obtain only shape variation all Euclidean transformations have been filtered out in Figure 4 (right) b.m.o. a Generalised Procrustes Analysis (GPA). Further, the Procrustes mean shape is drawn in red. To get a more clear picture of the variation of each landmark, its principal directions are plotted as ellipses in Figure 6 (left). This reveals clearly that the point variation is heteroscedastic in the plane. This is especially true for landmarks at the upper jaw and on the lips.

To obtain an impression of how correlated the landmarks are, refer to the matrix in Figure 5 (left). In this analysis shapes are laid out as $\mathbf{s} = [\ x_1\ \ldots\ x_N\ y_1\ \ldots\ y_N\ ]^{\mathrm{T}}$. Observe, for example that all x-positions of the left and right eye (landmarks 14–21 and 22–29) are highly correlated. A rotational invariant measure of correlation is shown in Figure 5 (right). This is the canonical correlation[4] between each landmark. Here the major five block diagonals are jaw, eyes, eyebrows, mouth and nose.

[4]The maximal correlation between two variables when general linear mappings on both are allowed.

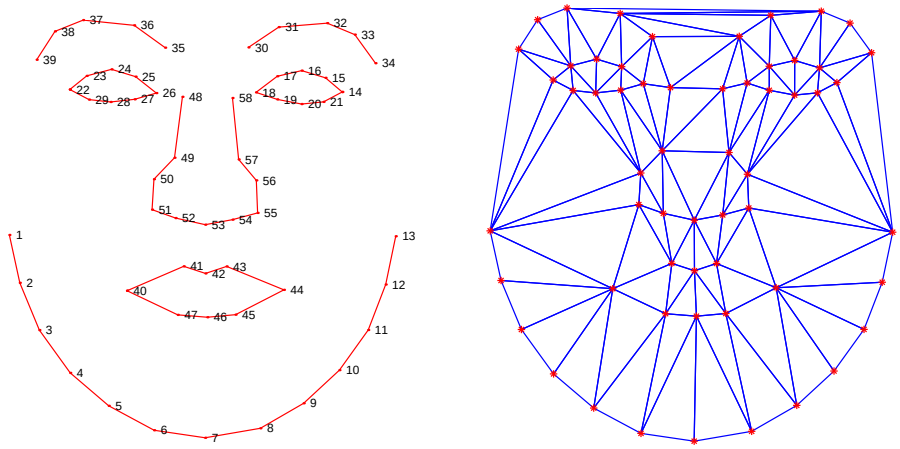


Figure 3: Facial landmarks (left). Triangular model mesh (right).

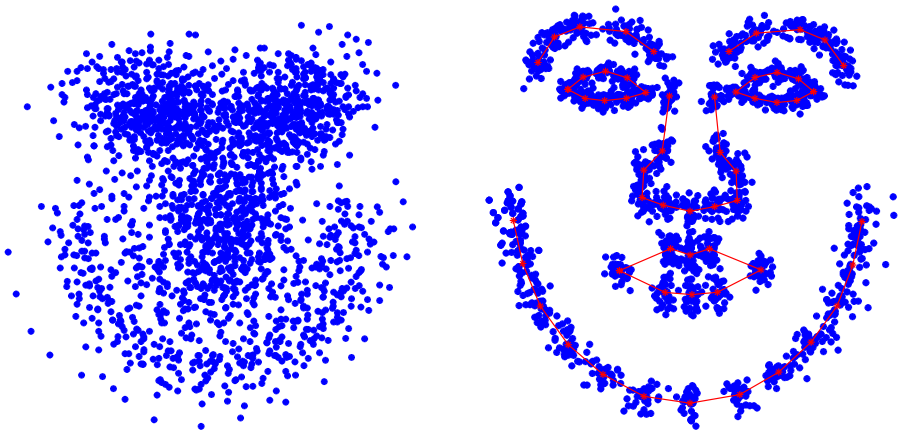


Figure 4: Unaligned shapes (left). Aligned shapes (right).

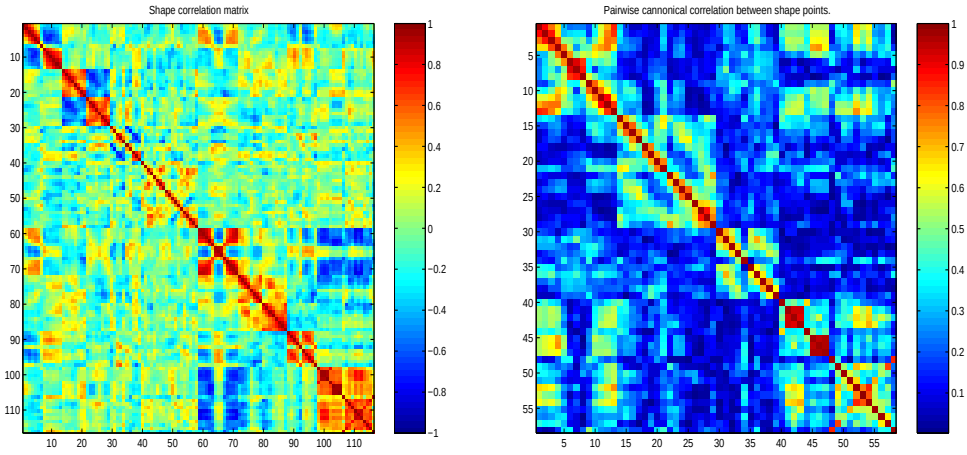


Figure 5: Shape correlation matrix (left). The canonical correlation of landmarks (right).

Moving to globally correlated point movements, the first shape mode (i.e. the first eigenvector of the covariance matrix) is shown in Figure 6. This is the direction in the subspace spanned by the 37 face annotations – embedded in a $2\times58=116$ dimensional space – with the highest variance. The most dominant deformation is the upward movement of the upper jaw together with a downward movement of nose and mouth. Since the

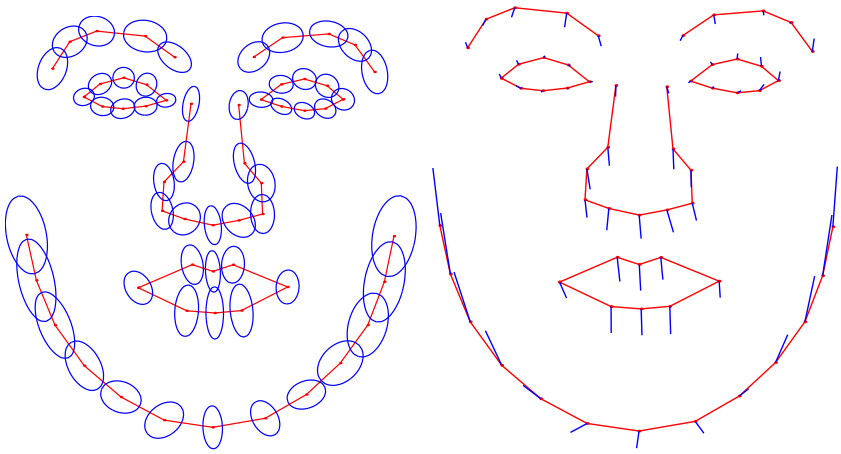


Figure 6: Principal point directions (left). First eigenvector as displacement vectors (right).

sign of the eigenvector can be arbitrary chosen these movements could equally well be in the inverse direction. When visualising this deformation as a movie sequence, it is clear that a major part of the variation picked up in this direction in hyperspace does not correspond to inter-suject variation in head anatomy. It is merely changes in the projection of the 3D landmarks into the 2D image plane, stemming from the head posture.

The amount of variance explained by the ten largest eigenvalues is shown in Table 2. Figure 7 (left) shows a plot of the 20 largest eigenvalues accounting for 95% of the total shape variation. The three largest shape modes are plotted as deformations of the mean shape in Figure 8.

To examine if any outliers are included into the shape model, all 37 faces are projected onto the first and second shape mode in Figure 7 (right). Here face number 28 is revealed as an outlier in principal component two. This is confirmed by comparing Figure 9 and 8. Finally the five largest principal scores are inspected for non-linearities in the scatter plot in Figure 10. Though not perfectly Gaussian, the PCA should still yield reasonable results on this distribution.

Table 2: Ten largest eigenvalues of the shape model

Mode	Variance	Acc. variance
1	37.34%	37.34%
2	12.66%	50.00%
3	8.22%	58.22%
4	5.92%	64.14%
5	4.64%	68.77%
6	4.32%	73.10%
7	3.45%	76.55%
8	2.69%	79.24%
9	2.43%	81.67%
10	2.18%	83.85%

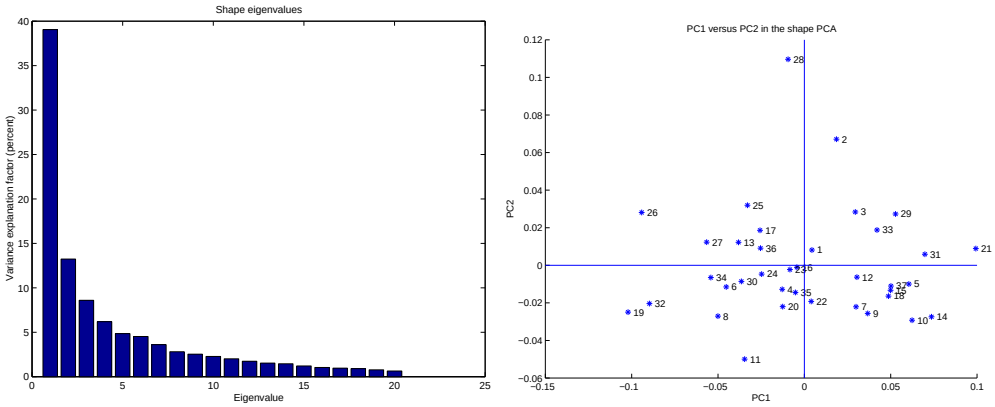
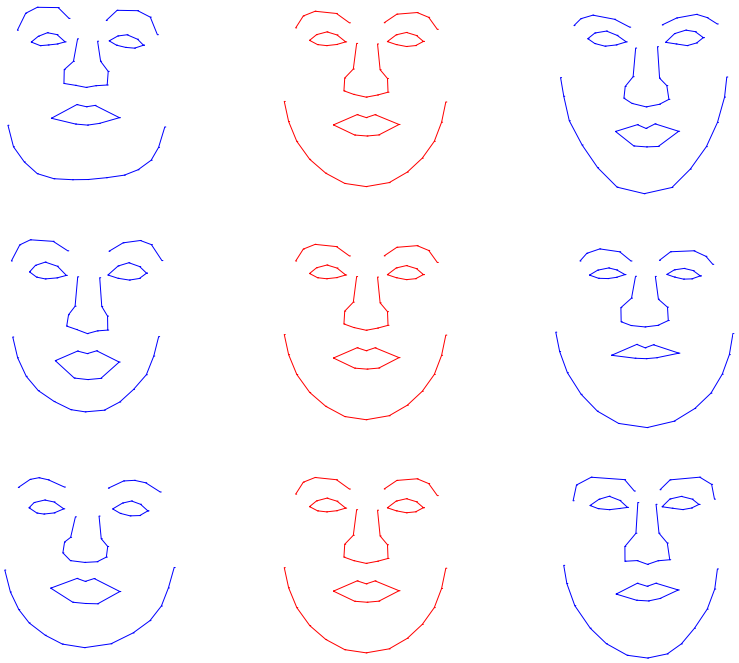


Figure 7: Shape eigenvalues (left). PC1 ($b_{s,1}$) vs. PC2 ($b_{s,2}$) in the shape PCA (right).



-3 std. dev. mean shape +3 std. dev.

Figure 8: Shape deformation using the three largest principal modes (row-wise, top-down).

Figure 9: Shape outlier, observation # 28.

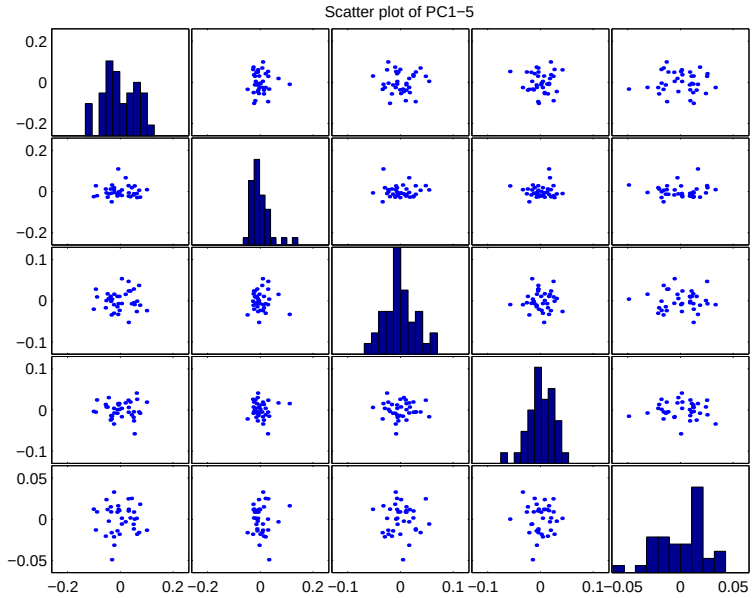


Figure 10: Scatter plot of the five largest shape principal components (PC1 is top-left).

5.2 Texture Model

The face texture model for all 37 faces is built from 31.221 sample positions in the reference frame, which is the Procrustes mean shape sized to mean size. The sample positions is obtained by sampling the reference frame in a one-pixel spaced grid aligned with the x and y axis. At each position a red, green and blue sample was obtained, resulting in a colour texture vector of 93.663 samples.

Table 3 shows the ten largest eigenvalues of the texture PCA model. The three largest of these are visualised as deformations of the mean texture in Figure 12, $\pm$ three standard deviations. Figure 11 (left) shows the 29 largest eigenvalues, accounting for 95% of the texture variation in the training set. To inspect the training set for texture outliers all 37 faces are projected onto the first and second principal axes in Figure 11 (right). From the first mode in Figure 12 we can use Figure 11 to determine the degree of "beardedness" in faces. See e.g. observation number 3 and 19 in Figure 13.

Table 3: Ten largest eigenvalues of the texture model

Mode	Variance	Acc. variance
1	19.80%	19.80%
2	9.58%	29.38%
3	7.61%	36.99%
4	6.52%	43.51%
5	5.69%	49.20%
6	4.71%	53.91%
7	4.15%	58.06%
8	3.50%	61.56%
9	3.29%	64.84%
10	2.96%	67.81%

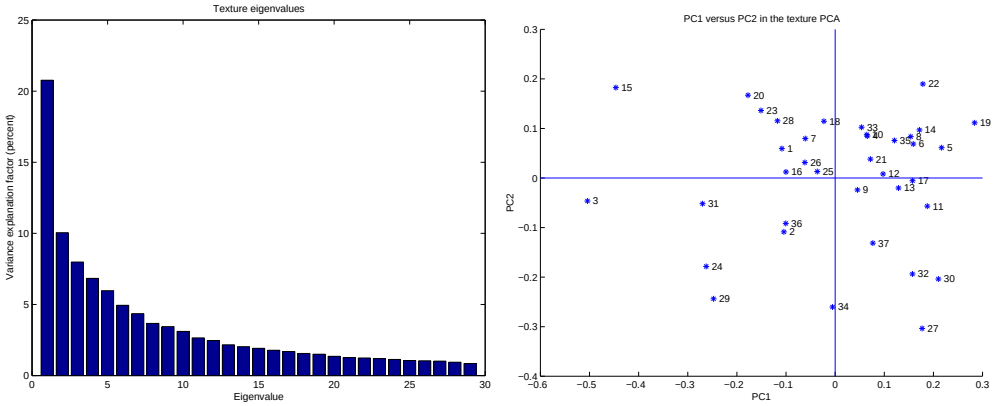


Figure 11: Texture eigenvalues (left). PC1 ($b_{s,1}$) vs. PC2 ($b_{s,2}$) in the texture PCA (right).

Finally, the diagonal of the covariance matrix is mapped onto the reference shape in Figure 14. This shows high variance at the nostrils and eyes and indirectly suggests that landmarks at the tip of the nose and at the pupils should be added in future face annotations. Further, some asymmetry is present at the upper jaw line, which could stem from a somewhat asymmetric lighting and/or inaccurate jaw annotation at the right-hand side (i.e. the participants left-hand side).

Figure 12: Texture deformation using the three largest principal modes (row-wise, top-down).

Figure 13: Extreme face textures. Min/max texture PC1 ($b_{t,1}$), obs. #3 (left), obs. # 19 (right).

Figure 14: Texture variance, black corresponds to high variance.

5.3 Appearance Model

Since AAMs use coupled eigenspaces of shape and texture, these PC scores are combined in a third PCA. To make the normalised measures of pixel distance and pixel intensities commensurate, the texture PC scores are weighted by the square root of the ratio between the texture and shape eigenvalues.

The ten largest eigenvalues are shown in Table 4. The corresponding three largest deformation modes of shape and texture are shown in Figure 15. Refer to the homepage mentioned in Section 2.2 to obtain the AAMExplorer for real-time exploration of the modes of the combined appearance model.

Table 4: Ten largest eigenvalues of the combined model

Mode	Variance	Acc. variance
1	22.74%	22.74%
2	12.59%	35.33%
3	7.82%	43.16%
4	5.81%	48.96%
5	5.17%	54.13%
6	4.29%	58.42%
7	4.00%	62.42%
8	3.42%	65.84%
9	3.14%	68.98%
10	2.94%	71.91%

-3 std. dev. mean +3 std. dev.

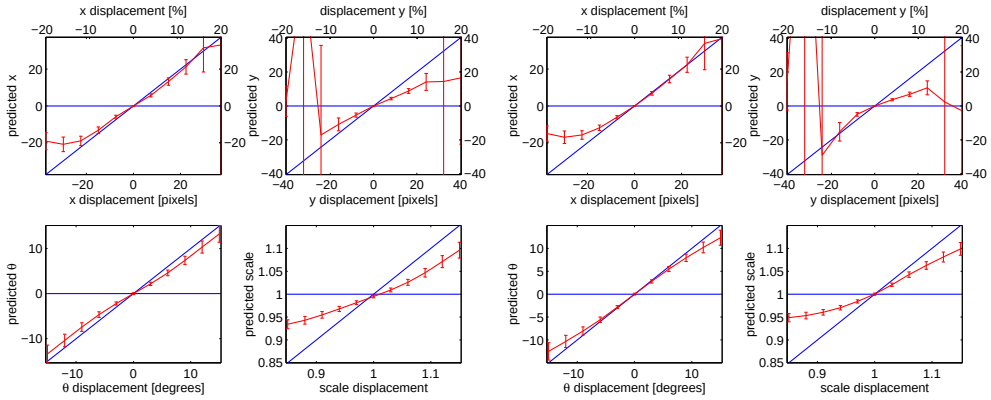
Figure 15: Combined deformation using the three largest principal modes (row-wise, top-down).

5.4 Model Training

In Section 3.1 two methods for model training were summarised. This section aims at assessing the quality of these methods for predicting AAM parameters. Both hypothesised a simple linear relationship between a residual vector and needed parameter updates. Desirable properties of such a prediction matrix include:

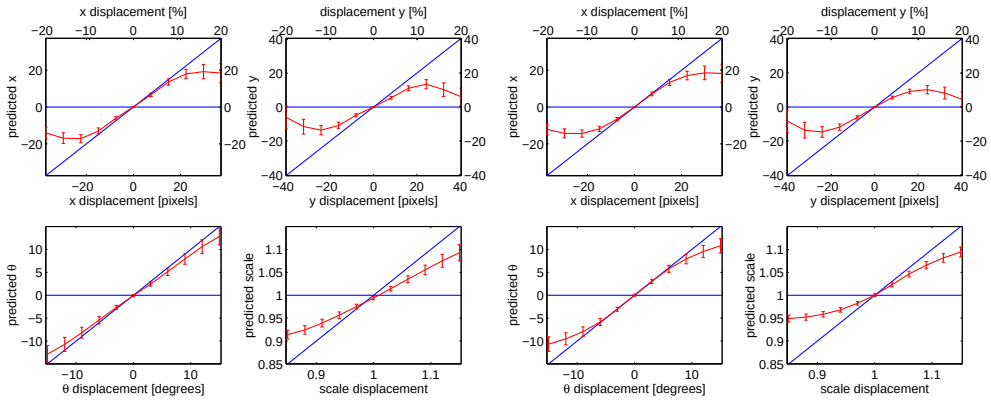
- Ability to predict the displacements learned
- Ability to interpolate and (even) extrapolate the displacements learned
- High prediction accuracy around zero

First we want the above properties to hold for the training set. Second we can hope for it to generalise to unknown images. To assess the pose prediction abilities, all 37 training shapes were systematically displaced in all pose parameters, one by one. Results for both learning methods in grey-scale and colour are shown in Figure 16. Error bars are one std. dev. In all cases only four images was used to train the models (the first image and then every tenth). It is interesting to notice the large discrepancy between the x and y prediction quality present for both learning methods using grey-scale data. In comparison the more specific colour models seems much more stable for large displacements in y, leading to models with larger convergence radius. In general, these plots provide no significant evidence for choosing one learning method over the other. Consequently, one should choose the Jacobian due to its substantial lower computational complexity.



(a) Grey-scale, regression

(b) Grey-scale, Jacobian



(c) Colour, regression

(d) Colour, Jacobian

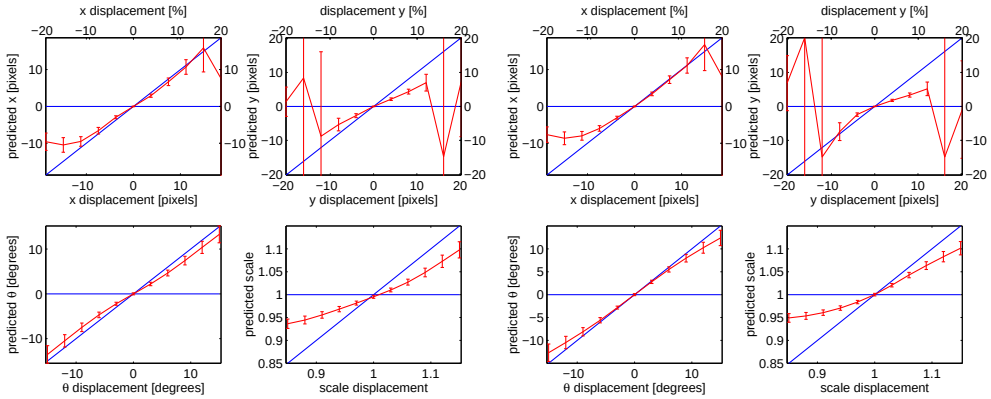
Figure 16: Prediction of pose parameter updates for large models.

To investigate the behaviour of the two learning methods at different scales we have decimated all face images to half size. The results using these are shown in Figure 17.

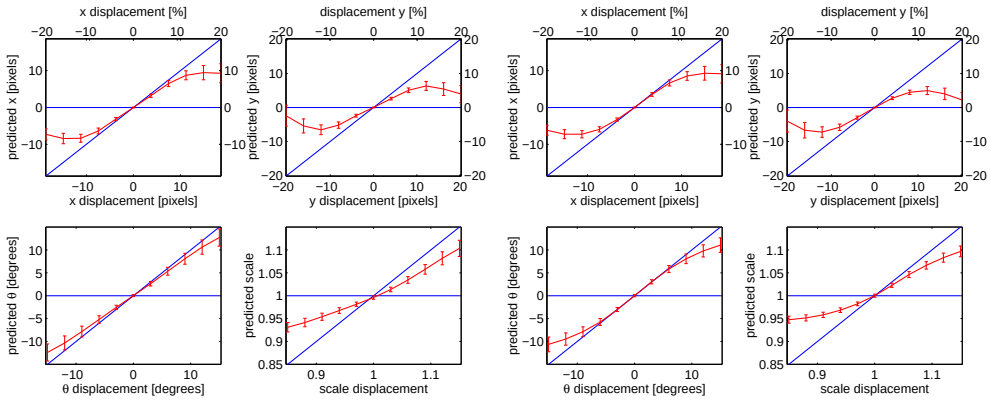
When measured relative to the shape size, we observe that the results are very similar to Figure 16. This contradicts the results given – on a different implementation and training set – in [4] where pose predictions at different scales deviates highly.

Due to the subsampling scheme Figure 16 and 17 and is a mixture of predictions upon known (11%) and unknown data (89%). What is left to investigate is how well these training methods generalise to unseen images with a mixture of displaced parameters. This is the topic of the next section.

Here we have restricted ourselves to investigating pose parameters. Further studies of parameter prediction should also include model parameter predictions.



(a) Grey-scale, regression **(b)** Grey-scale, Jacobian



(c) Colour, regression **(d)** Colour, Jacobian

Figure 17: Prediction of pose parameter updates for small models.

5.5 Segmentation Accuracy

Though the generative properties of AAMs enable very sophisticated image interpretation directly, the most common application remains to be segmentation.

To assess the segmentation accuracy the model was initialised using the mean configuration displaced 10% in x and y, relative to its width and height, i.e. four experiments per training example.

Due to the limited size of the training set, cross-validation was applied in a leave-one-out analysis leading to 37 models built from 36 examples for each evaluation. Double-mean landmark errors were calculated as the mean of all landmark points and mean of all leave-one-out experiments. Thus, $58 \times 4 \times 37 = 8584$ pt.pt. and pt.crv. distances were calculated in each evaluation. Pt.pt. measures the Euclidean distance between corresponding landmarks of the model and the ground truth. Pt.crv. measures the shortest

distance to the curve in a neighbourhood of the corresponding landmark.

Results for colour and grey-scale AAMs in two resolutions are shown in Table 5 and 6. AAMs in Table 6 were built in 1:2 decimated versions of the training images, i.e. 320×240 pixels. These models had on average 7.799 and 23.398 texture samples (grey-scale and colour, respectively). All shape, texture and combined PCA models were truncated to explain 95% of the variation present in the training set, resulting in 24 combined model parameters on average.

From both tables we observe that the Jacobian training scheme is (slightly) preferable w.r.t. segmentation accuracy. This is very important due to the far smaller computational and memory demands of the Jacobian training scheme compared to the regression training scheme. To our knowledge, this has never been empirically shown. But [2] has pointed towards this behaviour.

From Table 5 and 6 we also observe that the addition of colour to the models only resulted in a modest improvement of the final segmentation accuracy. However, colour added significant stability to the parameter update process as shown in Section 5.4. Also notice the fairly small penalty in segmentation accuracy when working on 1:2 decimated images.[5]

Table 5: A: Segmentation results – large models

Model Type	Learning Method	Mean pt.pt.	Mean pt.crv.
Grey-scale	Regression	6.24±1.36	3.08±0.88
Grey-scale	Jacobian	6.12±1.39	2.99±0.96
Colour	Regression	6.08±1.11	3.08±0.85
Colour	Jacobian	5.91±1.15	2.87±0.86

Table 6: Segmentation results – small models

Model type	Learning method	Mean pt.pt.	Mean pt.crv.
Grey-scale	Regression	3.30±0.86	1.63±0.53
Grey-scale	Jacobian	3.24±0.76	1.56±0.50
Colour	Regression	3.31±0.66	1.64±0.46
Colour	Jacobian	3.14±0.63	1.49±0.46

Due to the excessive memory consumption of the regression approach, the subsampling scheme in Section 5.4 was rather crude. But, recall that the memory usage of the Jacobian estimation does not depend on the number of training shapes. Therefore we have tested if even higher precision could be obtained using all training examples for the Jacobian training scheme. The results shown in Table 7 are pointing towards this behaviour, showing a subtle increase in accuracy.

Table 7: Segmentation results – large models, no subsampling

Model type	Learning method	Mean pt.pt.	Mean pt.crv.
Grey-scale	Jacobian	6.00±1.33	2.91±0.94
Colour	Jacobian	5.88±1.13	2.85±0.84

[5] When scaling the results for small models by a factor of two, that is.

Table 8: Segmentation results – small models, no subsampling

Model Type	Learning Method	Mean pt.pt.	Mean pt.crv.
Grey-scale	Jacobian	3.14±0.68	1.50±0.46
Colour	Jacobian	3.05±0.59	1.46±0.42

5.6 Model Truncation

Using parallel analysis as presented in Section 3.2 the shape and texture models can be truncated according to the scree plots shown in Figure 18. In this case the first 3 shape modes and the first 9 texture modes are selected. In contrast a 95% variance threshold would select 19 and 28 modes, respectively. From a computational point-of-view, one would prefer the simpler model, i.e. chose the parallel analysis solution. However, the truncation could prove to be to crude, i.e. resulting in under fitting. Another possibility is that the variance threshold is to liberal, resulting in over fitting. We have tested this b.m.o. a leave-one-out analysis and assessed the segmentation accuracy. The combined model was truncated at 95% variance as in the previous experiments.

The results given in Table 9 and 10 shows – in comparison with Table 7 and 8 – a modest increase in performance wrt. pt.pt. distance. This encouraging result suggest that parallel analysis provides both faster *and* more general AAMs in a non-parametric fashion. Consequently, simple variance thresholding seems to over fit the training data slightly by including to many modes of variation. It would be interesting to confirm this behavior on larger training sets.

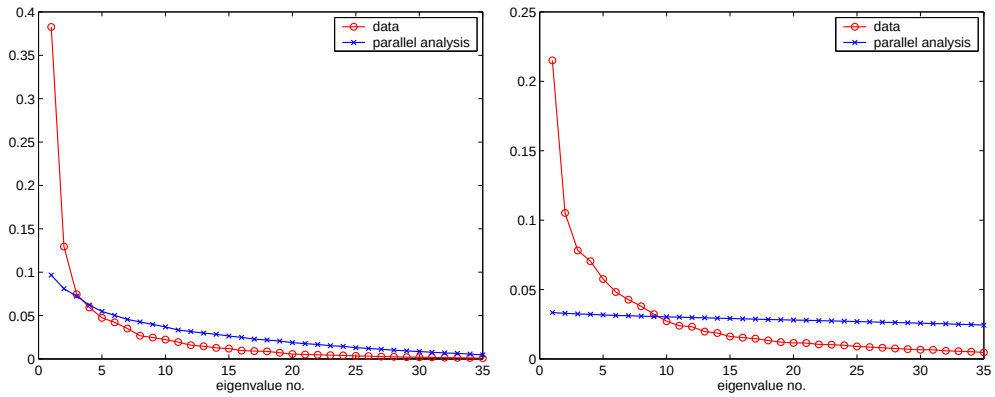


Figure 18: Scree plots for raw (red) and randomised data (blue). Shape (left), texture (right).

Table 9: Segmentation results – large models, no subsampling

Model type	Learning method	Mean pt.pt.	Mean pt.crv.
Grey-scale	Jacobian	5.74±1.18	3.04±0.75
Colour	Jacobian	5.54±1.18	2.93±0.81

Table 10: Segmentation results – small models, no subsampling

Model Type	Learning Method	Mean pt.pt.	Mean pt.crv.
Grey-scale	Jacobian	2.86±0.61	1.50±0.38
Colour	Jacobian	2.78±0.60	1.47±0.40

5.7 Details on Machinery and Computation Time

To give an impression of the applicability of these face models we acquired timings for the model building phase and the optimisation phase. These are meant only as reference numbers. If the purpose is to obtain maximal speed a multi-resolution model should be used. he build timings were calculated for both large and small models in colour and grey-scale built on all 37 training examples using the subsampling scheme mentioned in Section 5.4. All timings in Table 12 were obtained on similar models but using leave-one-out on the training examples.

All results in this report including the performance measures in Table 11 and 12 were obtained on an Athlon 1200 MHz equipped with 768 MB RAM using software warping. Except for the last row that used a GeForce 2 MX for hardware warping. From Table 11 and 12 we can observe that warping is not the bottleneck in the current implementation.

In Table 13 we used a Pentium 4 mobile 1700 Mhz equipped with an NVidia GeForce 4 Go graphics board. We have used high performance timers and several hundred repetitions in order to measure accurately in the millisecond range. From this table we can observe that the specific graphics board (and its driver) has a great impact on the performance. Further, since the hardware AAM warping is very heavy on bus i/o the infrastructure between GPU and CPU is of utmost importance.[6]

Table 11: Model building and warping (small/large models)

	Gray-scale	Colour
Model build, Regression	0:33 / 1:16 mins	0:56 / 2:45 mins
Model build, Jacobian	0:09 / 0:40 mins	0:26 / 1:54 mins
Software analysis warp	1 / 9 ms	3 / 12 ms
Hardware analysis warp	2 / 5 ms	2 / 7 ms

Table 12: Optimisation timings (small/large models)

	Gray-scale	Colour
Mean number of iterations	11.4 / 11.1	8.7 / 8.1
Mean optimisation time	137 / 545 ms	275 / 1240 ms
1 optimisation iteration	12 / 49 ms	32 / 153 ms

Table 13: Warp timings – GeForce 4 Go

	Gray-scale	Colour
Software analysis warp	5.4 ms	7.4 ms
Hardware analysis warp	1.5 ms	2.4 ms

6 Discussion

This report has treated many aspects and design choices regarding the building process of Active Appearance Models (AAMs). Using frontal face images, topics such as model training, model truncation, model resolution, et cetera have been covered and benchmarked using the segmentation capabilities of the respective models.

Parts of the presented results will generalise to other cases, while other parts probably won't. We therefore emphasise that all comments below only are based on the current face study. However, we hope that all of the presented results may serve as guidelines when building AAMs where thorough exploration of the design possibilities is not possible.

We have shown that the Jacobian training scheme is preferable in several ways. It requires little memory, it is simple to implement, faster to use and give results comparable

[6]The actual rendering is very fast compared to AGP data bus i/o.

to the regression approach (PCR). Further, it can be applied to larger training sets using more perturbations thus providing even better results than PCR.

Not surprisingly, the added specificity of colour models tended to enlarge the convergence radius and adding stability. However, when initialised inside the convergence radius of the equivalent grey-scale models, the final segmentation accuracy is very similar.

Non-parametric truncation of the shape and texture models has been carried out using parallel analysis. This lead to faster, more general and far simpler models with comparable and slightly increased performance (wrt. accuracy), when compared to conventional variance thresholding.

In conclusion and wrt. to this specific face study; a good compromise between speed and accuracy is obtained using grey-scale, AAMs built on 1:2 down-scaled images, trained using the Jacobian approach with no subsampling of the training set, and truncated using parallel analysis. If the highest accuracy is the goal, the above scheme should be used, but using the original 640×480 colour images.

Software and data have been made available to enable reproduction of all results provided by this report. Further, we have illustrated how to exploit the rapid development of graphics hardware to provide a fast implementation of the AAM framework. Finally, by providing the actual code, we hope that this will shed light on the AAM details that this report inevitably left out and give inspiration to further research in the field of generative modelling.

References

[1] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. In *Proc. European Conf. on Computer Vision*, volume 2, pages 484–498. Springer, 1998.

[2] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *IEEE Trans. on Pattern Recognition and Machine Intelligence*, 23(6):681–685, 2001.

[3] T. F Cootes and P. Kittipanya-ngam. Comparing variations on the active appearance model algorithm. In *Proceedings of the British Machine Vision Conference, BMVC*, volume 2, pages 837–846, 2002.

[4] T. F. Cootes and C. J. Taylor. *Statistical Models of Appearance for Computer Vision.* Tech. Report. Feb 2000, University of Manchester, http://www.isbe.man.ac.uk/∼bim/, 2000.

[5] T. F. Cootes and C. J. Taylor. Statistical models of appearance for medical image analysis and computer vision. In *Proc. SPIE Medical Imaging 2001*, volume 1, pages 236–248. SPIE, 2001.

[6] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Active shape models - their training and application. *Computer Vision and Image Understanding*, 61(1):38–59, 1995.

[7] T.F. Cootes and C.J. Taylor. Constrained active appearance models. *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, 1:748–754 vol.1, 2001.

[8] G. J. Edwards, C. J. Taylor, and T. F. Cootes. Interpreting face images using active appearance models. In *Proc. 3rd IEEE Int. Conf. on Automatic Face and Gesture Recognition*, pages 300–5. IEEE Comput. Soc, 1998.

[9] J. C. Gower. Generalized Procrustes analysis. *Psychometrika*, 40:33–50, 1975.

[10] J. L. Horn. A rationale and test for the number of factors in factor analysis. *Psychometrika*, 30:179–186, 1965.

[11] M. J. Jones and T. Poggio. Multidimensional morphable models: a framework for representing and matching object classes. *International Journal of Computer Vision*, 29(2):107–31, 1998.

[12] R. R. Paulsen, R. Larsen, S. Laugesen, C. Nielsen, and B. K. Ersbøll. Building and testing a statistical shape model of the human ear canal. In *Medical Image Computing*

and Computer-Assisted Intervention - MICCAI 2002, 5th Int. Conference, Tokyo, Japan,. Springer, 2002.

[13] S. Sclaroff and J. Isidoro. Active blobs. *Proc. of the Int. Conf. on Comput. Vision*, pages 1146–1153, 1998.

[14] M. B. Stegmann. Active appearance models: Theory, extensions and cases. Master's thesis, Informatics and Mathematical Modelling, Technical University of Denmark, Lyngby, 2000. http://www.imm.dtu.dk/∼aam/.

[15] M. A. Turk and A. P. Pentland. Face recognition using eigenfaces. In *Proc. 1991 IEEE Com. Soc. Conf. on CVPR*, pages 586–91. IEEE Com. Soc. Press, 1991.

Acknowledgements

We want to gratefully acknowledge the following people for their help in this work. The face database was built and partly annotated by Michael Moesby Nordstrøm, Mads Larsen and Janusz Sierakowski at the IMM computer vision laboratory. Dmitry Karasik made the multi-band extension of the AAM-API used in the colour experiments. Bent Dalgaard Larsen provided the laptop for the benchmarking results in Section 5.7. Finally, young Nikoline Hassing Stegmann kindly agreed in letting us use her picture for the introduction.

Appendix

A ASF – AAM Shape Format Specification

An ASF file is structured as a set of lines separated by a CR character. Anywhere in the file, comments can be added by starting a line with the '#' character. Comment lines and empty lines are discarded prior to parsing. The layout of an ASF file is as follows:

- Line 1 contains the total number of points, n, in the shape.
- Line 2 to n+1 contains the point information (one line per point) such as the point location, type, connectivity etc., see below. Hence, quick and simple access is preferred over data compactness.
- Line n+2 contains the *host image*, i.e. the filename of the image where the annotation is defined.

The formal point definition is:

point := <path#> <type> <x-pos> <y-pos> <point#> <connects from> <connects to>

<path#> The path that the point belongs to. Points from different paths must not be interchanged (in the line order).

<type> A bitmapped field that defines the type of point:

- Bit 1: Outer edge point/Inside point
- Bit 2: Original annotated point/Artificial point
- Bit 3: Closed path point/Open path point
- Bit 4: Non-hole/Hole point

Remaining bits should be set to zero. An inside artificial point which is a part of an closed hole, has thus the type: $(1\text{<<}1) + (1\text{<<}2) + (1\text{<<}4) = 1 + 2 + 4 = 7$.

<x-pos> The relative x-position of the point. Obtained by dividing image coordinates in the range [0;image width-1] by the image width (due to strange historic reasons...). Thus, pixel $x = 47$ (the 48th pixel) in a 256 pixel wide image has the relative position 47/256 = 0.18359375.

<y-pos> The relative y-position of the point. Obtained by dividing image coordinates in the range [0;image height-1] by the image height (due to strange historic reasons...). Thus, pixel $y = 47$ (the 48th pixel) in a 256 pixel tall image has the relative position 47/256 = 0.18359375.

<point#> The point number. First point is zero. This is merely a service to the human reader since the line at where the point occurs implicitly gives the real point number.

<connects from> The previous point on this path. If none <connects from> == <point#> can be used.

<connects to> The next point on this path. If none <connects to> == <point#> can be used.

Further, the following format rules apply:

- Fields in a point specification are separated by spaces or tabs.
- Path points are assumed to be defined clockwise. That is; the outside normal is defined to be on left of the point in the clockwise direction. Holes are thus defined counter-clockwise.
- Points are defined in the fourth quadrant. Hence, the upper left corner pixel is (0,0).
- Isolated points are signaled using <connects from> == <connects to> == <point#>.
- A shape must have at least one outer edge. If the outer edge is open, the convex hull should determine the interior of the shape.

Example ASF file

```
<BOF>
######################################################################
#
#    AAM Shape File  -  written: Monday May 08 - 2000 [15:22]
#
######################################################################

#
# number of model points
#
83

#
# model points
#
# format: $<$path#$>$ $<$type$>$ $<$x rel.$>$ $<$y rel.$>$ $<$point#$>$ $<$connects from$>$ $<$connects to$>$
#
0  0  0.07690192  0.44500541  0  82  1
0  0  0.09916206  0.42914406  1  0  2
0  0  0.12925033  0.39573063  2  1  3

...

0  0  0.07579006  0.52910086  80  79  81
0  0  0.06128729  0.49762829  81  80  82
0  0  0.05858913  0.46610570  82  81  0

#
# host image
#
F1011flb.bmp

<EOF>
```

B Hardware-assisted AAM Warping

Contemporary graphics hardware is highly optimised for performing piece-wise affine warps. Below we will demonstrate how to exploit this in an AAM framework.

In AAMs two types of warps are carried out. One during the analysis phase (i.e. model building and model optimisation) and a one during synthesis. In the analysis phase many-to-one warps are carried out. Different configurations of the shape model are all warped to same shape free reference frame, which is typically the mean shape sized to mean size. During synthesis shape free textures are warped to specific shape configurations.

We approach the graphics hardware through the industry standard for graphics programming: OpenGL. Here a triangular mesh can be defined and images can be projected onto this surface b.m.o. texture mapping. The steps involved in both analysis and synthesis cast in an OpenGL setting is enumerated below.

Analysis

1. The image that is going to be analysed is uploaded as a texture to the graphics board.
2. The current shape is uploaded as texture coordinates for the reference mesh, which is a Delaunay triangulation of the mean shape.
3. The reference mesh is rendered in an orthogonal projection where one unit corresponds to one pixel. Bilinear texture filtering is used.
4. The rendered image is downloaded to main memory.
5. Using a lookup table, the raster image of the reference mesh is converted to a texture vector by masking out background pixels.

Synthesis

1. Using a lookup table, a texture vector is converted into is equivalent shape free image.
2. The shape free image is uploaded as a texture.
3. An optional image can be uploaded and rendered as background for the synthesised AAM configuration.
4. The vertices in the reference mesh are uploaded as texture coordinates for a mesh where the current shape constitute the vertices.
5. The above mesh is rendered using bilinear texture filtering.
6. The rendered image is downloaded to main memory.

OpenGL Comments

The recent possibilities of having hardware-accelerated off-screen rendering buffers is used (the so-called *pbuffers*). This renders all usage of the graphics board invisible – and most important, uncontrollable – to the user of the AAM package. Further, if available we exploit the also recent possibilities of having non-dyadic textures.[7] If this extension is not available, images are zero-padded to width and height that are powers of two. This will naturally induce significant overhead in texture transfers.

Currently, the main bottlenecks are the texture and pbuffer transfers. Due the asymmetric design of the APG data bus, the former is relatively fast while the latter is very slow. Further, graphics boards and their drivers have also been optimised for writes rather than read, as this reflects the typical usage in CAD and gaming. However, with the advent of techniques within computer graphics such as dynamic texturing etc. this is very likely to change in the near future.

The use of techniques such as vertex arrays, compiled vertex arrays and the recent vertex array range extension are pleasing from a theoretical point of view. This is due to the set of constant vertices and texture coordinates during analysis and synthesis, respectively. However, since AAM meshes often are very small (i.e. few vertices) the effect of such techniques are negligible compared to time spent transferring textures and the pbuffer itself.[8] We have therefore settled with the intermediate mode in OpenGL when transferring geometry. Refer to the code for further details of the OpenGL usage.

[7] As implemented by the OpenGL extension `WGL_NV_render_texture_rectangle`.
[8] Using `glReadPixels()`.

C Face images

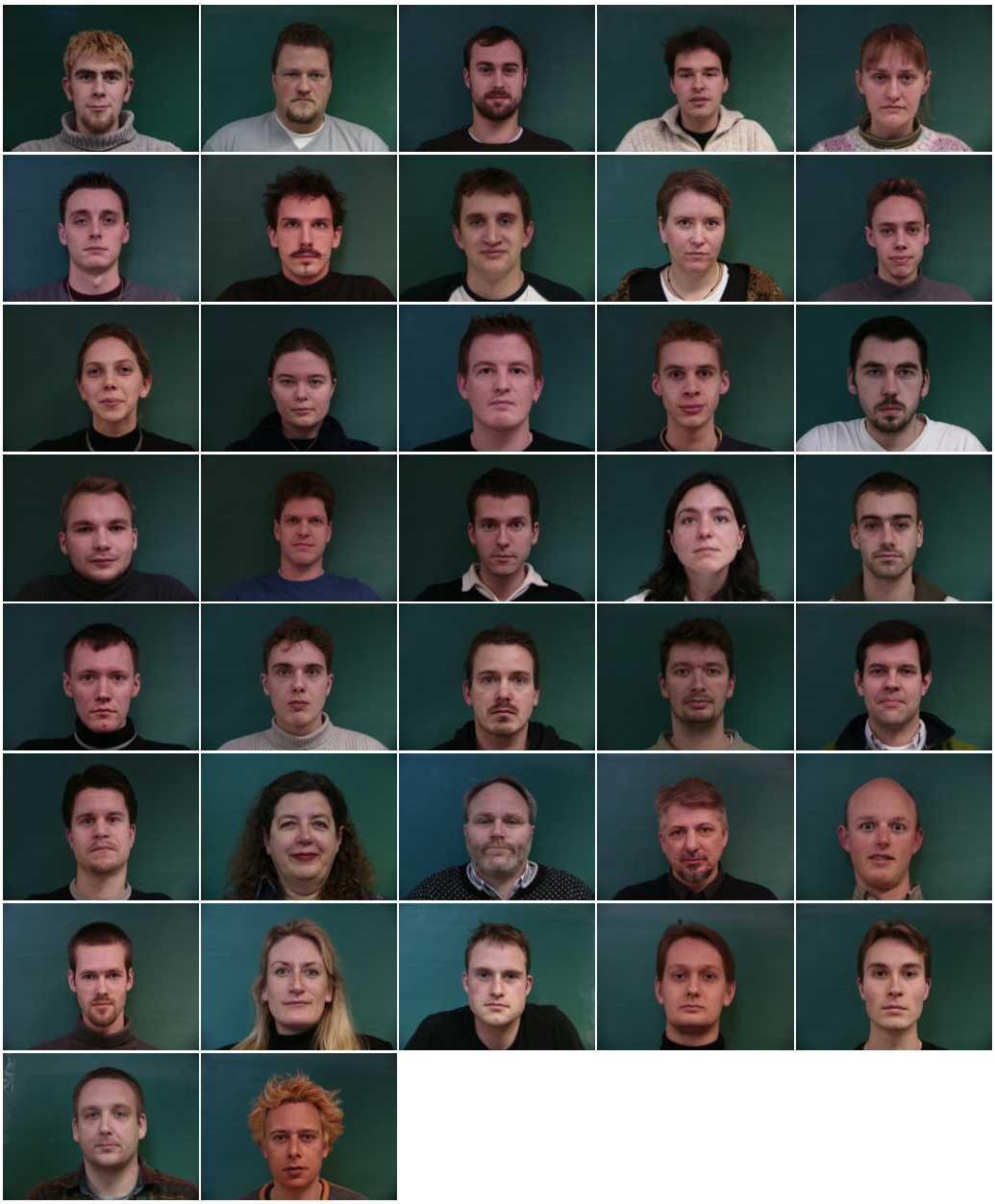


Figure 19: All 37 face images presented row-wise, top-down.